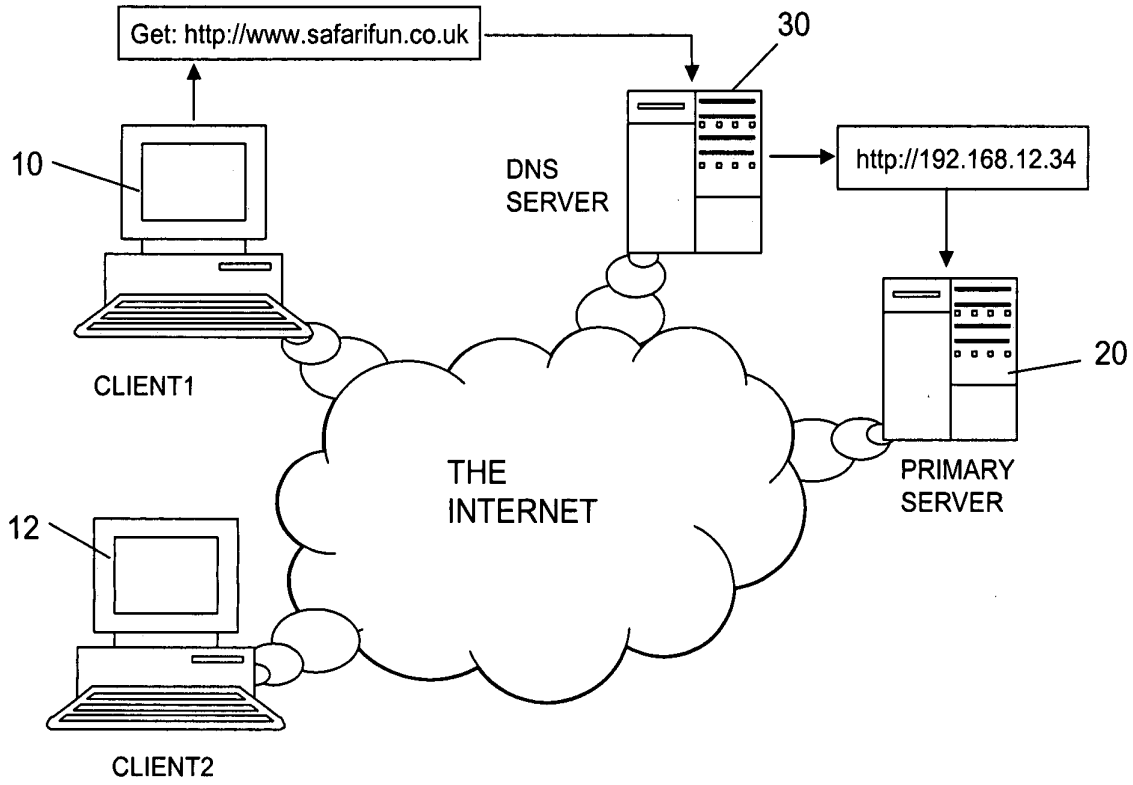

Get: http://www.safarifun.co.uk
30
10
DNS
SERVER
http://192.168.12.34
20
CLIENT1
THE
INTERNET
PRIMARY
SERVER
12
CLIENT2

Fig. 1

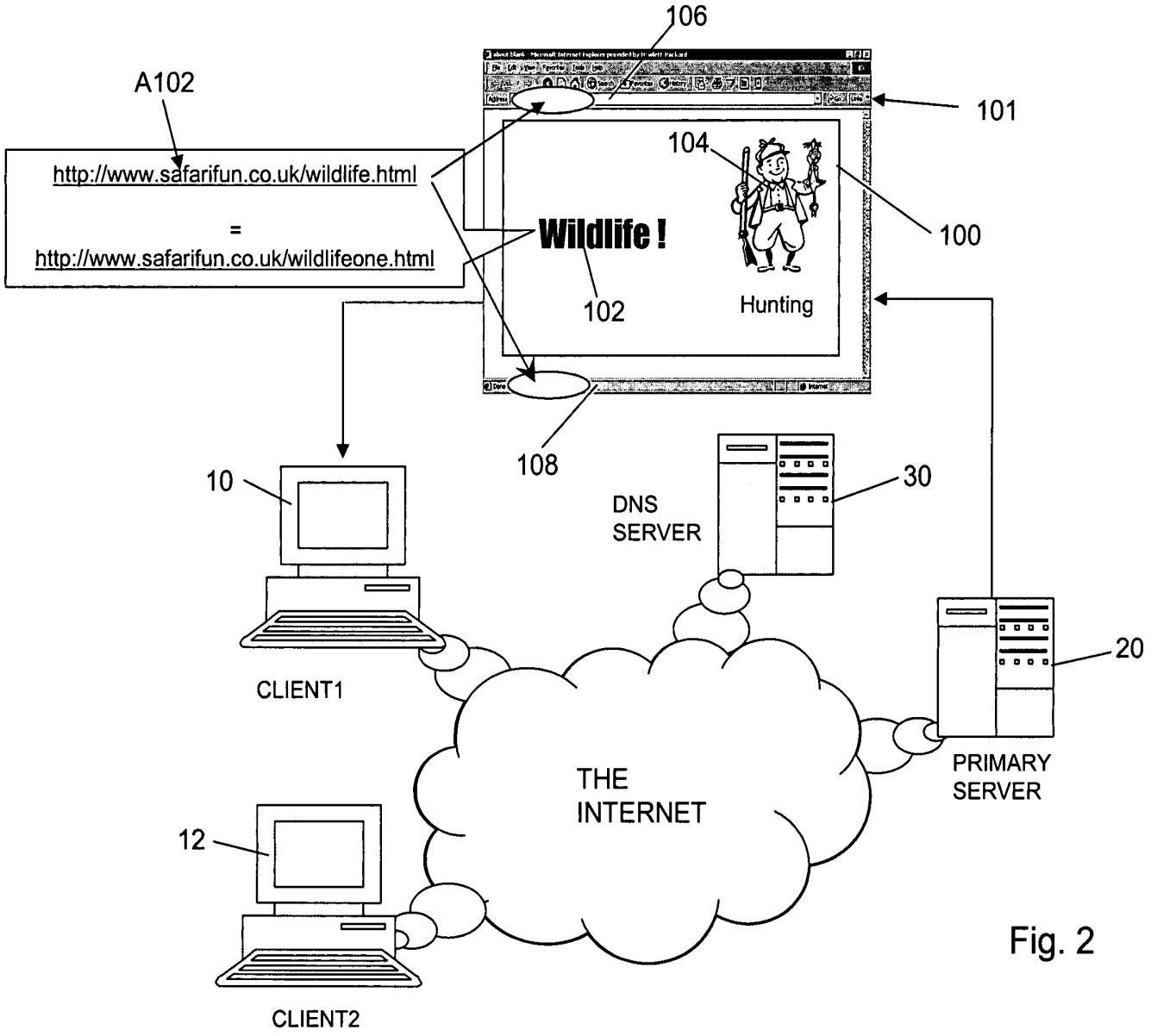


Fig. 2

PDNO 300201984

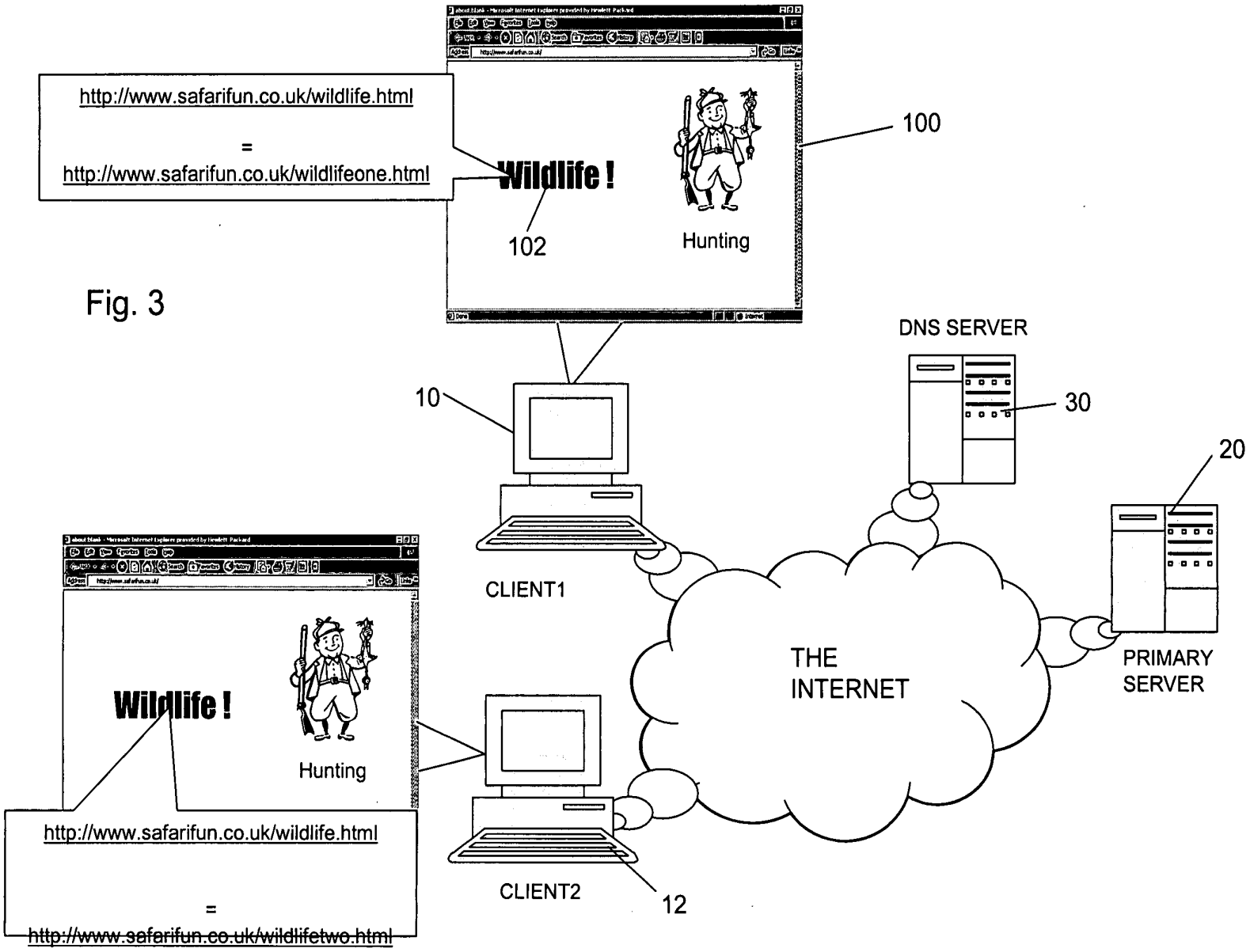
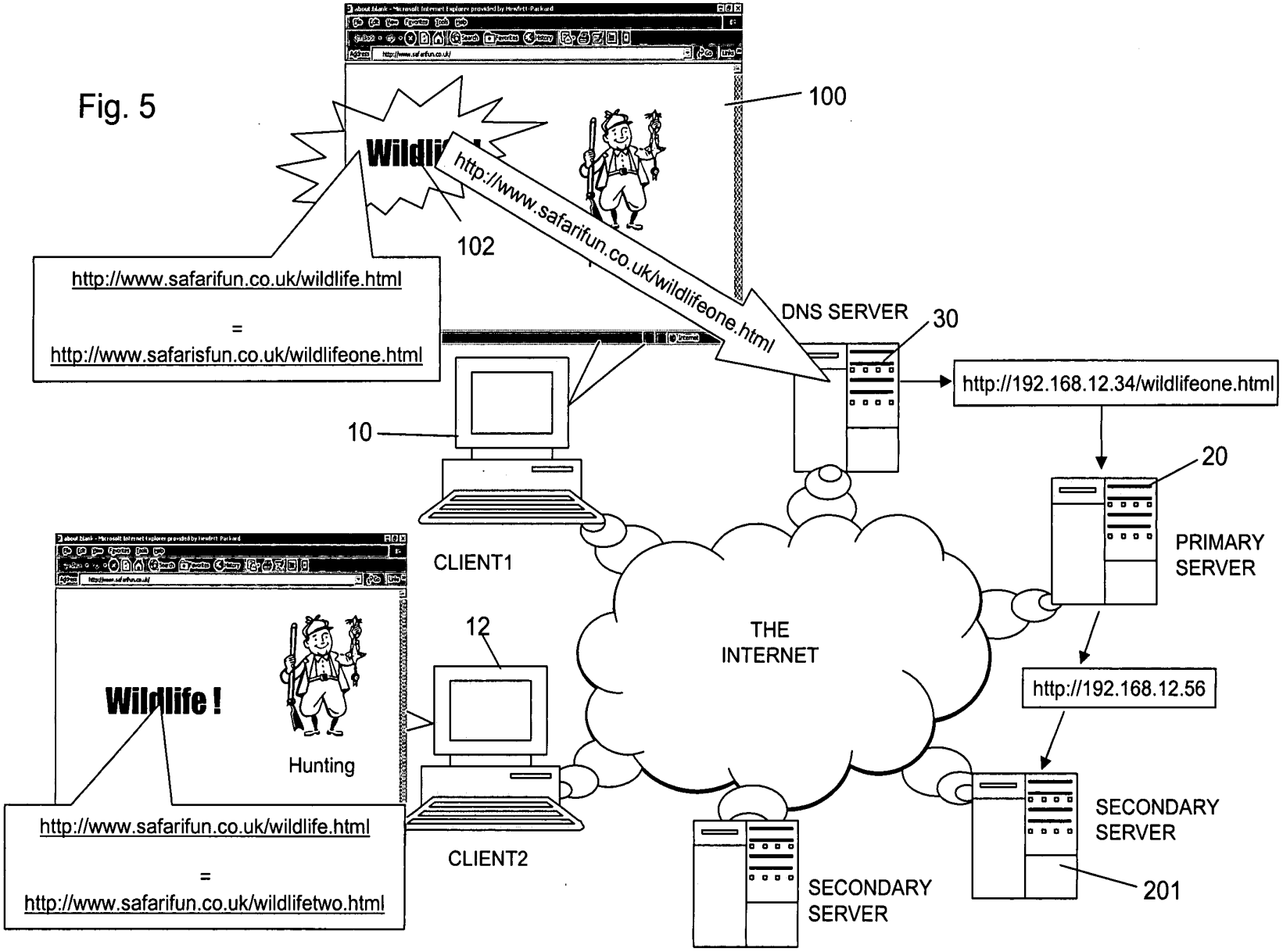

http://www.safarifun.co.uk/wildlife.html
=
http://www.safarifun.co.uk/wildlifeone.html
Wildlife !
Hunting
100
102
DNS SERVER
10
30
20
CLIENT1
THE
INTERNET
PRIMARY
SERVER
Wildlife !
Hunting
http://www.safarifun.co.uk/wildlife.html
=
http://www.safarifun.co.uk/wildlifetwo.html
CLIENT2
12

Fig. 3

```
<HTML>
<HEAD>
<TITLE>Safari Fun</TITLE>
<SCRIPT LANGUAGE="JavaScript">
  //
  // a simple alias based upon one common javascript implementation
  // this could also be acomplished through zero height frames with
  // page loaded into upper/lower frame
   function go_to_alias(){
     window.name="Safari Fun";
     window.navigation.bar="www.safarifun.co.uk";
     window.goto("http://www.safarifun/wildlifeone.html");
   }
</SCRIPT>
</HEAD>
<BODY>
  <SCRIPT LANGUAGE="JavaScript">
   // now call go_to_alias
   go_to_alias();
  </SCRIPT>
</BODY>
</HTML>
```

Fig. 4

Fig. 5



PDNO 300201984

Fig. 6

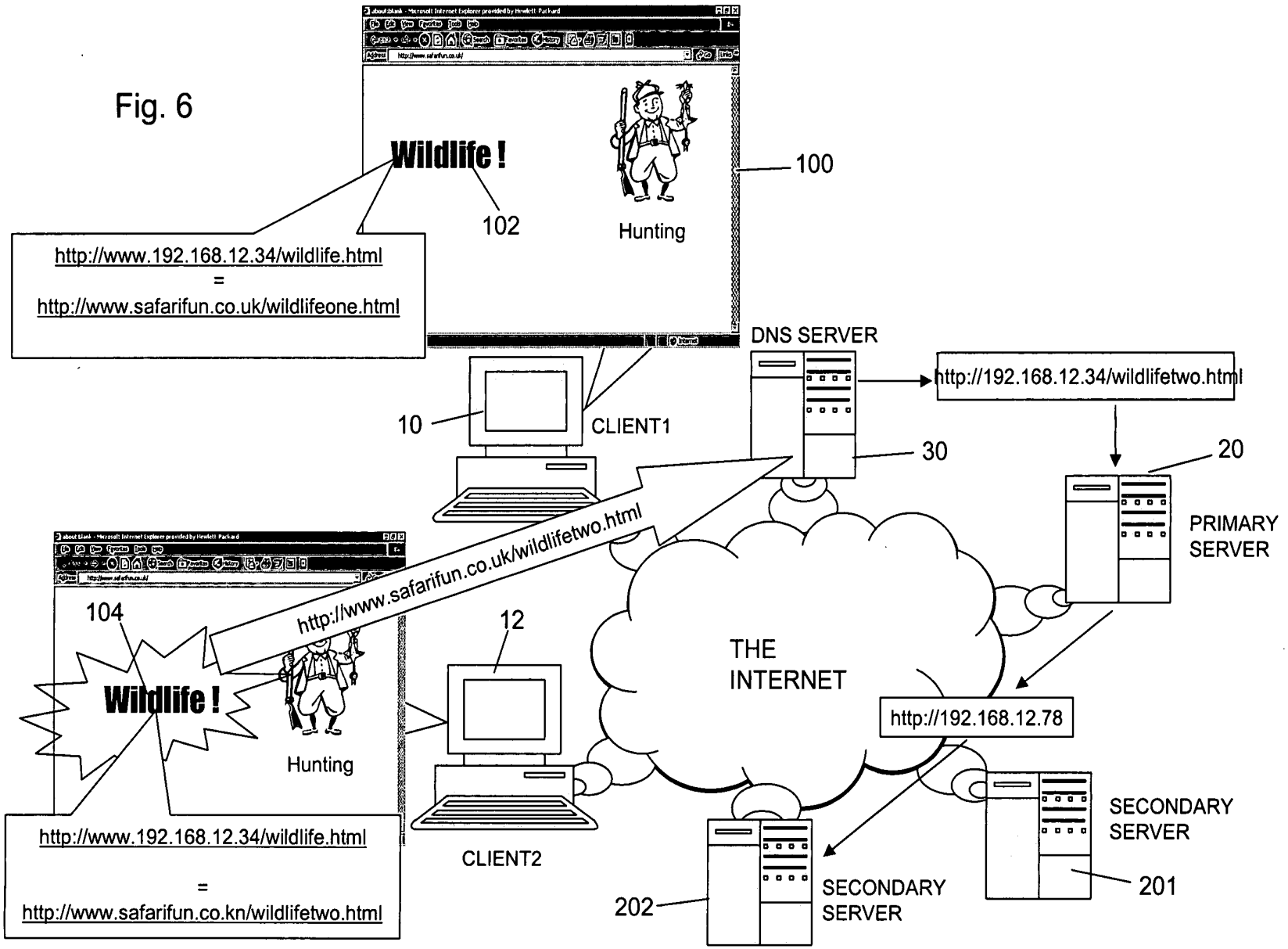

Wildlife !
Hunting
100
102
http://www.192.168.12.34/wildlife.html
=
http://www.safarifun.co.uk/wildlifeone.html
DNS SERVER
http://192.168.12.34/wildlifetwo.html
10
CLIENT1
30
20
PRIMARY
SERVER
http://www.safarifun.co.uk/wildlifetwo.html
104
12
THE
INTERNET
Wildlife !
Hunting
http://192.168.12.78
CLIENT2
SECONDARY
SERVER
202
SECONDARY
SERVER
201
http://www.192.168.12.34/wildlife.html
=
http://www.safarifun.co.kn/wildlifetwo.html

Fig. 7

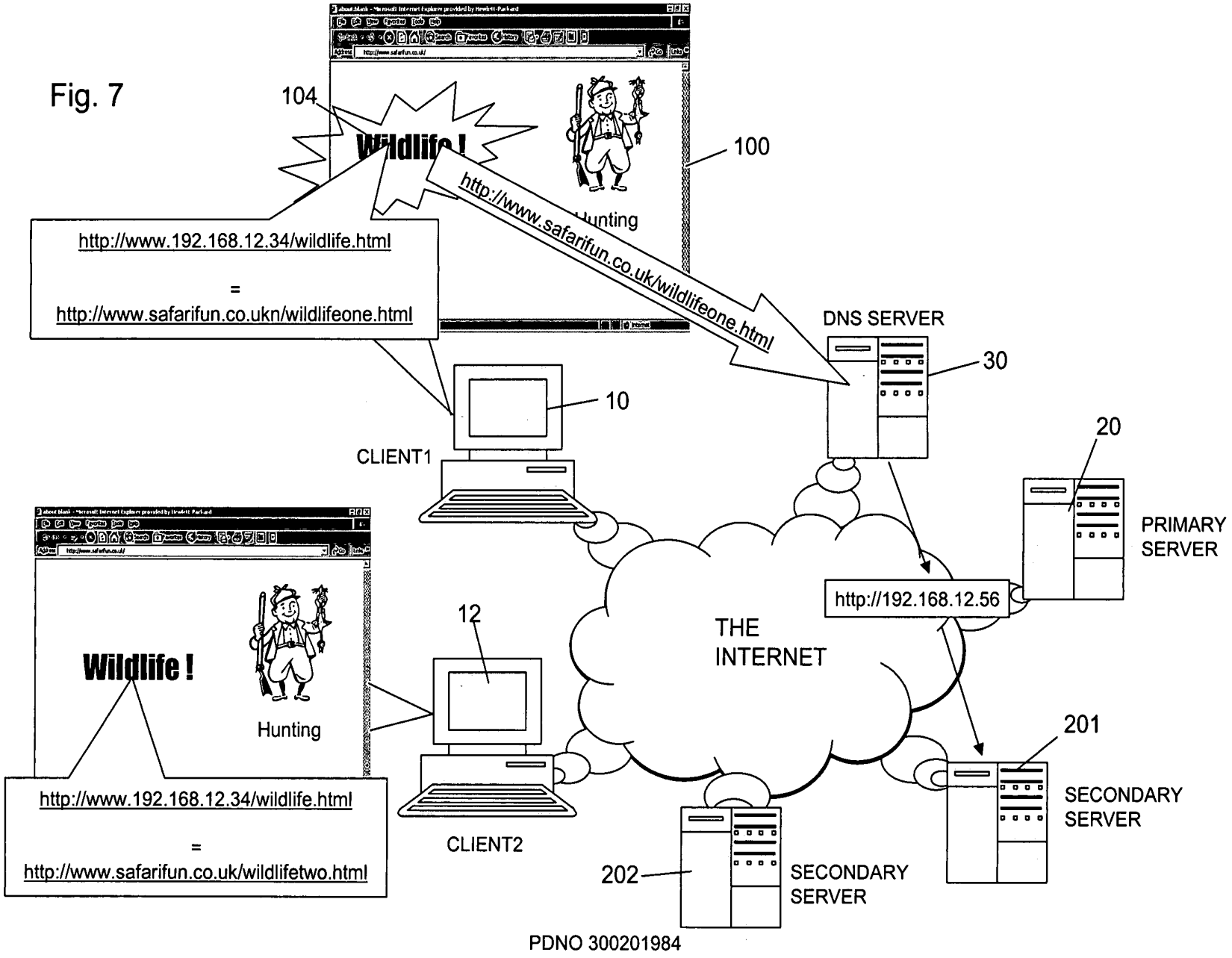
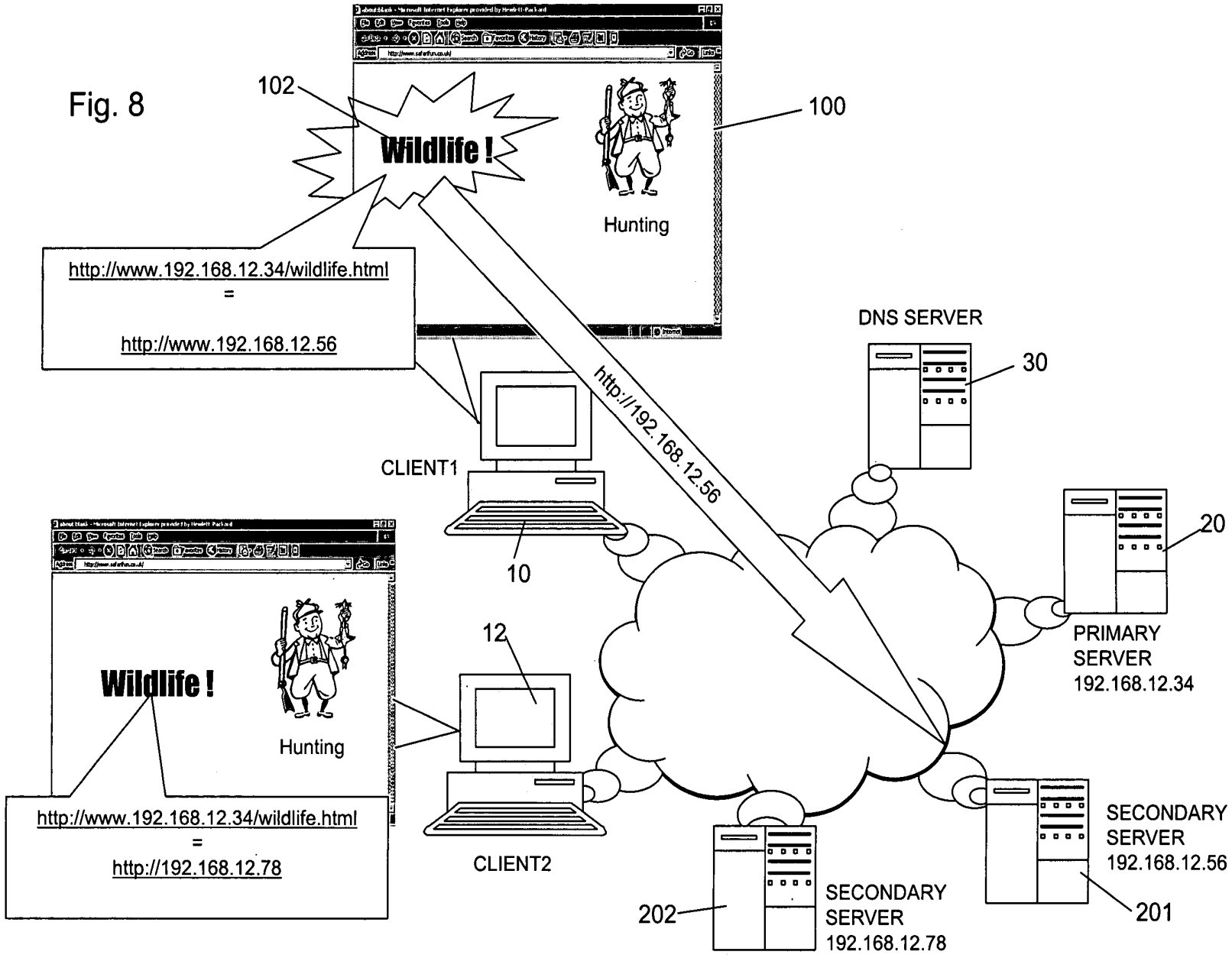

104
Wildlife !
Hunting
100
http://www.safarifun.co.uk/wildlifeone.html
http://www.192.168.12.34/wildlife.html
=
http://www.safarifun.co.ukn/wildlifeone.html
DNS SERVER
30
10
CLIENT1
20
PRIMARY SERVER
http://192.168.12.56
THE INTERNET
12
Wildlife !
Hunting
http://www.192.168.12.34/wildlife.html
=
http://www.safarifun.co.uk/wildlifetwo.html
CLIENT2
201
SECONDARY SERVER
202
SECONDARY SERVER
PDNO 300201984

Fig. 8


102
Wildlife !
Hunting
100
http://www.192.168.12.34/wildlife.html
=
http://www.192.168.12.56
DNS SERVER
30
http://192.168.12.56
CLIENT1
10
20
PRIMARY
SERVER
192.168.12.34
12
Wildlife !
Hunting
http://www.192.168.12.34/wildlife.html
=
http://192.168.12.78
CLIENT2
SECONDARY
SERVER
192.168.12.56
201
202
SECONDARY
SERVER
192.168.12.78

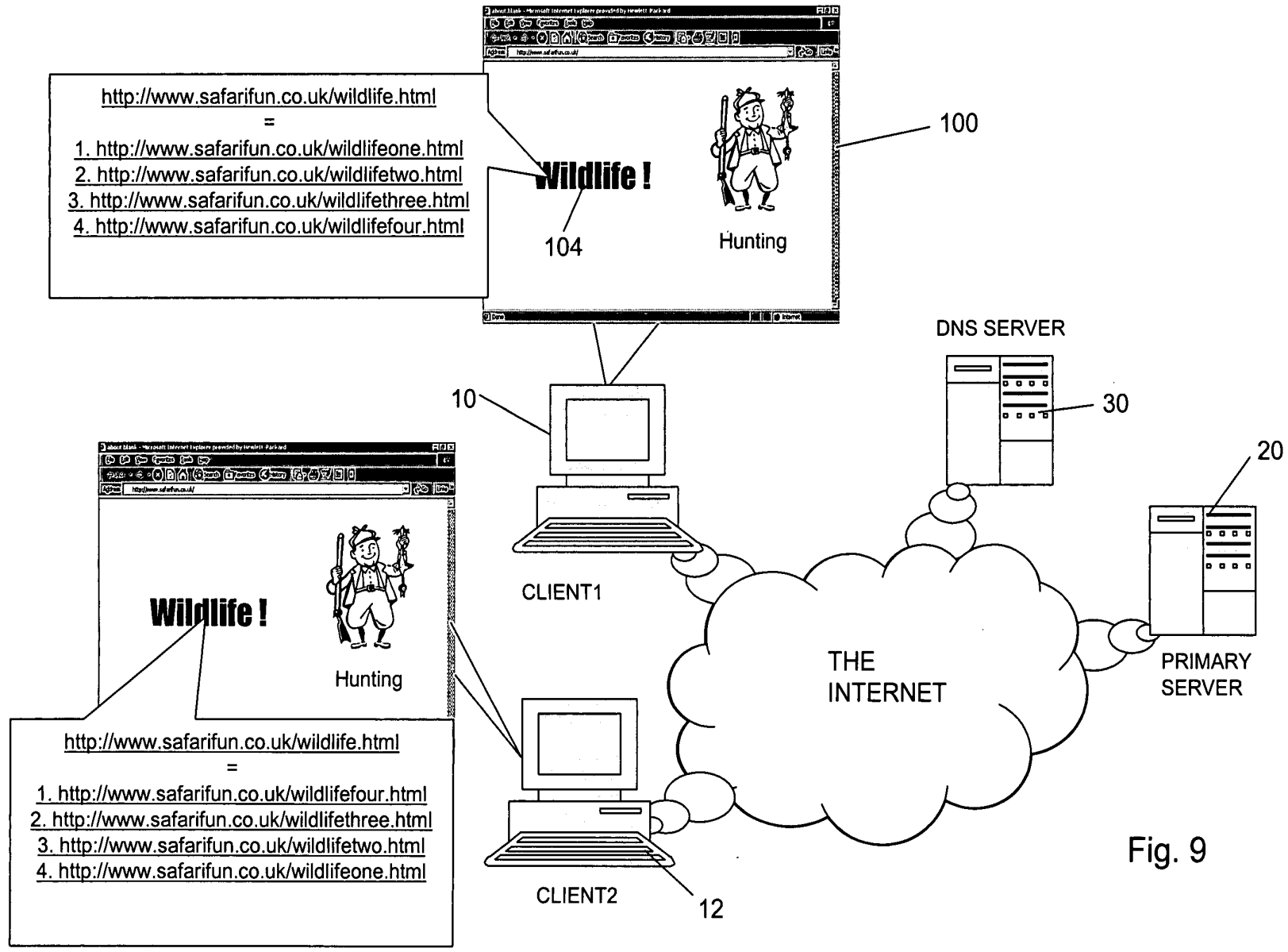

http://www.safarifun.co.uk/wildlife.html
=
1. http://www.safarifun.co.uk/wildlifeone.html
2. http://www.safarifun.co.uk/wildlifetwo.html
3. http://www.safarifun.co.uk/wildlifethree.html
4. http://www.safarifun.co.uk/wildlifefour.html
Wildlife !
Hunting
104
100
DNS SERVER
30
20
10
CLIENT1
THE
INTERNET
PRIMARY
SERVER
Wildlife !
Hunting
http://www.safarifun.co.uk/wildlife.html
=
1. http://www.safarifun.co.uk/wildlifefour.html
2. http://www.safarifun.co.uk/wildlifethree.html
3. http://www.safarifun.co.uk/wildlifetwo.html
4. http://www.safarifun.co.uk/wildlifeone.html
CLIENT2
12

Fig. 9

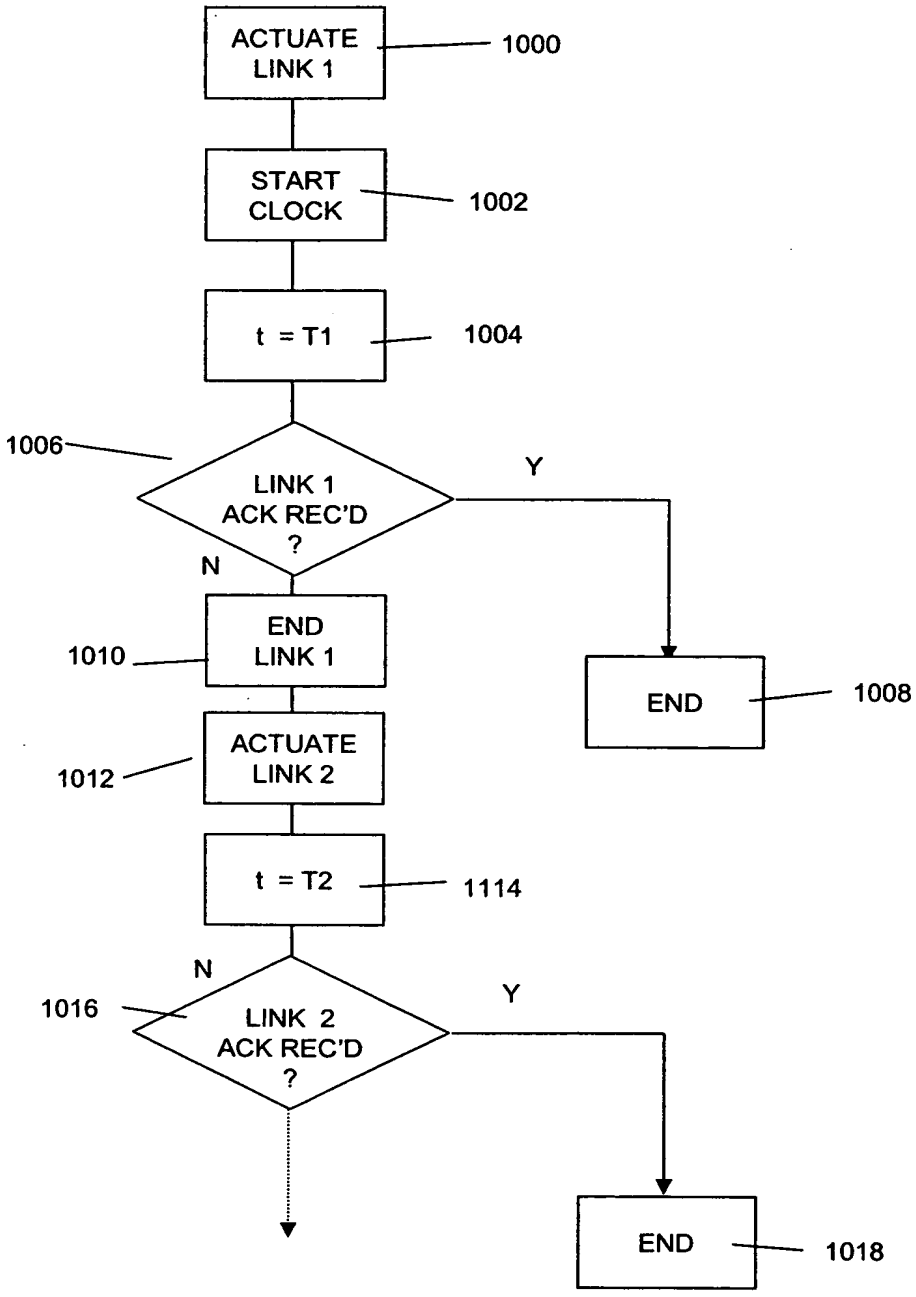

ACTUATE LINK 1
1000
START CLOCK
1002
t = T1
1004
1006
LINK 1 ACK REC'D ?
Y
N
END LINK 1
1010
END
1008
ACTUATE LINK 2
1012
t = T2
1114
N
1016
LINK 2 ACK REC'D ?
Y
END
1018

Fig. 10

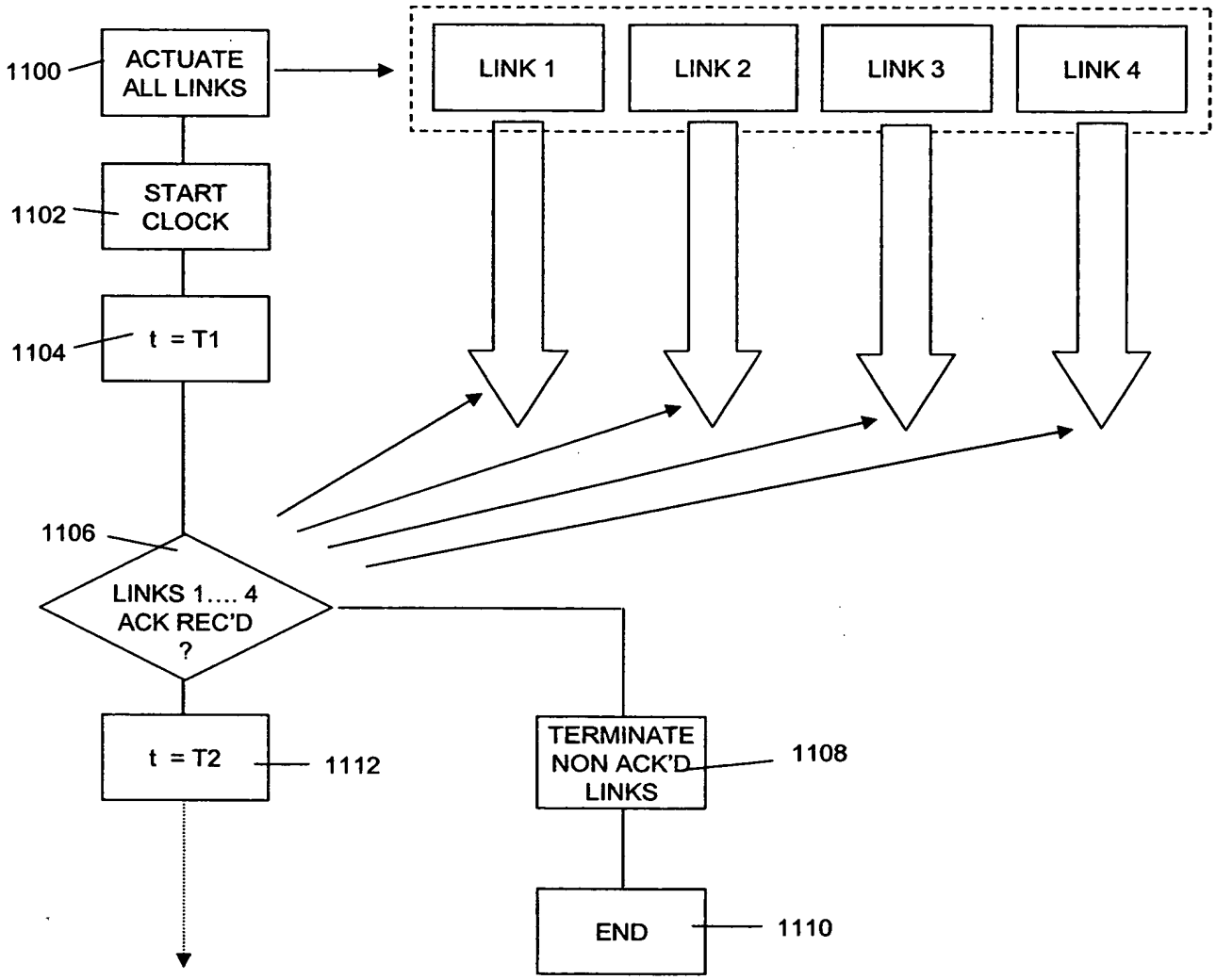

1100
ACTUATE ALL LINKS
LINK 1
LINK 2
LINK 3
LINK 4
1102
START CLOCK
1104
t = T1
1106
LINKS 1.... 4 ACK REC'D ?
t = T2
1112
TERMINATE NON ACK'D LINKS
1108
END
1110

Fig. 11

```
//
// takes as a parameter an array of strings, each of which
// represents a seperate link to try.
//
// executes links in order of presentation and returns successful
// (or none)note : returns string

public class testSLinks
{

  public static String main(String links[], int link_count, int time_out)
  {
      // default return behaviour
      String success_link = "";
      // for each link

      for (int i = 0; i < link_count; i++)
      {
          // fire up a 'getLink' function - not a thread  which terminates
          //on either time_out seconds (no response) or
             testThread(link[i], time_out, success_link).start();
      }
      // termination either leaves successful link in success_link
         or blank eturn(success_link);

  }
}
```

Fig. 12

```
//
// takes as a parameter an array of strings, each of which
// represents a seperate link to try.  Executes an individual
// thread to access each link.  One of three things happens:
// no threads complete in time_out; one thread completes
// continues with successful link killing off other threads
// still running; two or more threads complete - effectively a
// link is chosen to be loaded in a non deterministic manner
// (the result of java serialisation), kills off other threads
//  still running.
// note: returns string
// note: uses p_threads libaray to allow continuation as soon
// as one thread completes;

public class testPLinks
{

  public static String main(String links[], int link_count, int time_out)
  {
      // default return behaviour
      String success_link = "";
      // for each link

      for (int i = 0; i < link_count; i++)
      {
          // fire up a 'getLink' thread which terminates on either
          // time_out seconds (no response) or
             new testThread(link[i], time_out, success_link).start();
      }
      // termination either leaves successful link in success_link
         or blank return(success_link);
  }
}
```

Fig. 13